**Naredbe grananja**

**00:22:16**

Jedinica: 5 od 19

**Rezime**

* if else strukture omogućavaju uslovno izvršavanje nekog bloka koda;
* Ukoliko uslovno izvršeni blok ima više od jedne linije, treba da bude u vitičastim zagradama (u bloku);
* Uslovni izraz if else strukture mora biti boolean tipa;
* Moguće je uslovno dodeliti vrednost uz pomoć ternarnog operatora;
* if else strukture je moguće gnezditi jednu u drugu;
* Ako hoćemo da jednoj if else strukturi dodamo još jednu, koristimo if else if strukturu;
* Operatori koji se koriste u uslovnim izrazima su operatori poređenja.

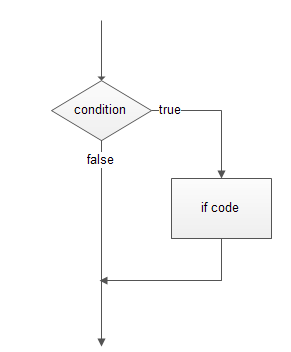
U ovoj lekciji bavićemo se uslovnim izvršavanjem, odnosno naredbama grananja. Upravljačke strukture omogućavaju rukovođenje tokom izvršenja programa. Na osnovu njihove strukture grupisaćemo ih u uslovne ili naredbe grananja, i one koje se ponavljaju a pritom predstavljaju programske petlje. Ako želite da obrada podataka koje je korisnik uneo ima smisla u daljem radu, potrebno je da vaš kod donosi odluke.

Naredbe unutar koda se uglavnom izvršavaju redom, onako kako su napisane. Nekad je neophodno promeniti tok izvršavanja programa. [Naredbe grananja](https://www.link-elearning.com/linkdl/opisPojma.php?id=139116) se koriste u slučajevima kada je na osnovu tačnosti nekog uslova potrebno izvršiti jedan ili drugi deo koda.

**Uslovno grananje – if**

Prva i osnovna naredba grananja je naredba if. Ova naredba poznata je u praktično svim jezicima i predstavlja baš ono kako se i zove: ako.

Kako reč „ako“, sama za sebe, nema baš mnogo logike u retorici, tako je nema ni u programiranju, već je potrebno da, osim nje, postavimo i neki uslov od koga će zavisiti ishod ove naredbe.



*5.1. if šema*

Kao što možemo videti na šemi iznad, program se izvršava dok ne stigne do dela koji izaziva grananje (if), gde se najpre proverava uslov. Ukoliko je uslov ispunjen (true), dolazi do izvršenja definisanog koda (if code). Ukoliko uslov nije ispunjen (false), preskače se if code (ne izvršava se), već se nastavlja sa učitavanjem strane.

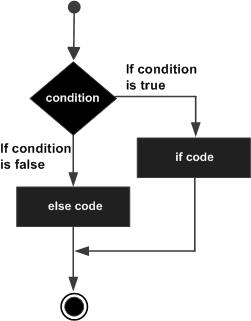
if-else može se kretati od osnovnog, minimalnog uslova:

if (uslov)  
      naredba;

koji predstavlja samo jedan uslov koji je potrebno ispuniti da bi se izvršila određena naredba. Preko mogućnosti alternativnog rešenja:

if (uslov)  
      { naredba  }  
else  
      { naredba2  };

Često će se desiti da je unapred potrebna odluka da li će se neka akcija uopšte izvršiti, ali potrebnije je odrediti koja grupa od više ponuđenih akcija treba da se izvrši baš u datom trenutku. Kada je ovakva problematika u pitanju, koristimo naredbu **else**u nastavku if uslovne naredbe. Naredba else se izvršava ukoliko uslov nije ispunjen.



*5.2. if else šema*

Dakle, ukoliko je uslov ispunjen (true), izvršiće se if blok koda (if code) i preskočiti alternativni blok (else code). Sa druge strane, ukoliko uslov nije ispunjen (false), odmah se prelazi na alternativni blok koda (else code) tako da blok if koda (if code) biva preskočen.

Pogledajmo sintaksu ove šeme:

if (*uslov koji se testira*) {  
*blok koda koji ce biti izvršen ako je uslov ispunjen*}else{  
*alternativni blok koda*    }

Kao što vidite, nakon zatvaranja if bloka koda (zatvorena vitičasta zagrada }), postavljamo ključnu reč **else**, nakon koje definišemo alternativni blok koda.

Evo kako uslovljavanje izgleda u praksi. Recimo da imamo jednu promenljivu $x koja ima vrednost 10. Ako hoćemo da prikažemo poruku samo ukoliko $x ima vrednost 10, napisaćemo:

|  |  |
| --- | --- |
| 1  2 | $x=10;  if($x==10) echo "x is 10"; |

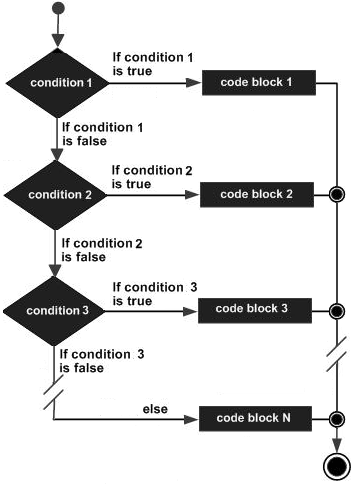
Napomena: Kao što se može videti u prethodnom primeru, kada je nakon provere uslova (koji je vratio true) potrebno izvršiti samo jednu liniju koda, nije neophodno tu liniju postavljati unutar vitičastih zagrada. To znači da bi isti primer, bez problema, mogao biti napisan i ovako:

          $x=10;  
         if($x==10){  
                  echo "x is 10";  
         }

**if...elseif....else naredbe**

Ova naredba, u opštem slučaju, ima sledeću sintaksu:

if (uslov1) {  
     naredbe 1  
}  
elseif (uslov2) {  
     naredbe 2  
}  
else {  
     naredbe 3  
}

**

*5.3. if elseif šema*

Razjasnimo ponašanje ovakve strukture. Najpre se proverava prvi uslov, i ukoliko je taj uslov ispunjen, izvršava se **code block 1**, a nakon toga se napušta struktura. Samo ukoliko prvi uslov nije ispunjen, prelazi se na testiranje drugog uslova. Ukoliko je drugi uslov ispunjen izvršava se njegov blok koda i napušta se struktura. Isto se ponavlja i za treći uslov. Na kraju, u slučaju da nijedan od dva uslova nije ispunjen, izvršava se alternativni (else) blok koda.

Evo kako bismo to mogli napisati:

if (prvi uslov) {  
         blok koda koji se izvršava ako je prvi uslov ispunjen  
}elseif(drugi uslov){  
         blok koda koji se izvršava ako je drugi uslov ispunjen  
}elseif(treći uslov){  
         blok koda koji se izvršava ako je treći uslov ispunjen  
}else{  
         alternativni blok koda koji se izvršava ako nijedan od uslova nije ispunjen  
}

Unutar jedne *if...else* strukture može biti više *elseif* blokova. Ukoliko je nekoliko uslova tačno, izvršiće se samo prvi od njih. Očigledno je da je ono sa čime rukuje mehanizam za uslovljavanje boolean tip, pa je tako jasno da u izraz nije moguće smestiti nešto čiji rezultat nije ovaj tip. Ali, ako se sećate iz prethodnih lekcija, PHP je u stanju da implicitno konvertuje tipove, tako da čak i ako bi izraz uslova bio samo broj ili aritmetička operacija, PHP bi bio u stanju da to adekvatno interpretira i postavi uslov.

Naredbe elseif možete da pišete i spojeno i odvojeno (else if), jer su oba oblika ispravna i u okviru primera ćete nailaziti na oba načina pisanja.

Kada koristite ovakav način pisanja koda, treba da imate u vidu da će samo jedan blok (jedna naredba koja je zadata po uslovu) biti izvršen. Uslovi mogu biti uzajamno isključivi (samo jedan može biti izvršen od svih navedenih), međutim, ako su uslovi takvi da više od njih može biti istovremeno ispunjeno, izvršava se samo jedna naredba ili [blok](https://www.link-elearning.com/linkdl/opisPojma.php?id=139091) naredbi iza prvog ispunjenog uslova.

**Ugnežđeni uslovi**

Ipak, sa semantičke strane, stavljanje brojne vrednosti u izraz uslova nije pravilno, pa to izbegavajte.

Na primer, sledeći primer je potpuno ispravan sa stanovišta funkcionalnosti programa, ali bi u praksi bolje bilo da i broj 10 ne bude broj, već neka promenljiva.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | <?php  $number = 15;  if ($number < 10) {      echo "Number is less than 10. ";  }  elseif ($number == 10) {      echo "Number is equal 10. ";  }  else {      echo "Number is greater than 10. ";  }  ?> |

Izjave grananja koda je moguće postavljati i jedne unutar drugih, odnosno ugnezditi ih. Na primer:

|  |  |
| --- | --- |
| 1  2  3 | if(true)      if(true)              echo "This line of code is executed"; |

Ovaj (potpuno funkcionalan) primer, osim što prikazuje ugnežđenje, prikazuje i još jedan način korišćenja ovog operatora, a to je telo bez vitičastih zagrada. Ovakav pristup moguć je samo ako telo podrazumeva jednu liniju koda. Ukoliko postoji više linija, bar dve, vitičaste zagrade su obavezne, inače može doći do logičkih nepravilnosti u radu aplikacije. Pogledajmo primer:

|  |  |
| --- | --- |
| 1  2  3 | if (true)      echo "This line of code depends of condition <br>";      echo "This line of code will be executed in any case"; |

Druga linija će u svakom slučaju biti izvršena. Što je u primeru banalan problem, ali u ozbiljnijoj aplikaciji može dovesti do značajnih grešaka.

Preporuka je da se vitičaste zagrade koriste i kada imate samo jednu naredbu nakon bloka da biste bili sigurni da vaša naredba pripada baš tom uslovu.

Pristup pisanja uslovnih blokova bez vitičastih zagrada sretaćete i kod drugih struktura za kontrolu toka, ali ne i u funkcijama ili klasama. Osim toga, obično će ono što ste zamislili kao jednu (uslovnu) liniju koda eskalirati na više linija, pa je možda najbolje, prilikom rukovanja tokom, držati se pristupa sa vitičastim zagradama.

Kada se javi potreba za složenijim logičkim operacijama, postoji mogućnost smeštanja jedne if naredbe u druge.

if (uslov1){  
blok koda 1;  
}else{  
if (uslov2){  
blok koda 2;  
}else{  
blok koda 3;  
}  
}

Analizirajmo ovaj slučaj. Ukoliko je **uslov1** ispunjen, izvršiće se **blok koda 1**. Ukoliko uslov nije ispunjen, spoljašnje grananje prelazi na else blok koda, u kojem se nalazi unutrašnje grananje. Ukoliko je **uslov2** koji je napisan u unutrašnjem grananju ispunjen, izvršava se **blok koda 2**. Ukoliko ni ovaj uslov nije ispunjen, izvršava se **blok koda 3**.

Ponekad se za uslovljavanje koristi i **ternarni** operator. Ovaj operator se koristi uglavnom za uslovnu dodelu vrednosti i ne sadrži blokove koda, tako da i nije baš prava struktura za kontrolu toka, već više operator dodeljivanja. Ipak, vrednost se dodeljuje uslovno, pa ovaj operator utiče na tok aplikacije.

|  |  |
| --- | --- |
| 1 | $b = ( $a == 0 ) ? 10 : $a; |

Ternarni operator podrazumeva četiri dela:

* + Promenljivu kojoj se dodeljuje vrednost ($b);
  + Uslov *($a==0);*
  + Vrednost koja se dodeljuje ako je uslov ispunjen (10);
  + Vrednost koja se dodeljuje ako uslov nije ispunjen ($a).

Recimo da negde ranije u kodu postoji promenljiva $a koja ima vrednost nula. Kada se aktivira ternarni operator, programu smo zapravo rekli da proveri neki uslov i dali mu alternative za oba ishoda tog uslova **? ishod 1**za slučaj da je uslov ispunjen i **: ishod 2** za slučaj da nije. Ternarni operator iz primera bi bio alternativa ovakvoj if strukturi:

|  |  |
| --- | --- |
| 1  2  3  4  5 | if($a == 0){      $b = 10;  }else{      $b = $a;  } |

U sledećem primeru se najpre definišu tri promenljive (day, month i year) koje sadrže podatke o datumu. Nakon toga se uz pomoć funkcije checkdate() proverava da li je datum validan. Grananjem koda određujemo koja će poruka biti prikazana na strani.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | $day = 15;      $month = 2;      $year = 2013;        $result = checkdate( $month, $day, $year);        if ($result == true)      {          echo "Date is correct.";      }      else      {          echo "Date is incorrect.";      } |

|  |
| --- |
| **Napomena**    Kako se u zagradi koja se nalazi nakon ključne reči if očekuje boolean vrednost, ona ne mora nastati kao posledica poređenja, već može biti upisana kao literal (konkretna vrednost). Tako smo u prethodnom primeru umesto izraza:    $result == true  mogli jednostavno napisati:    $result |

Konačno, evo i najčešće greške u kreiranju uslovnih blokova. Korišćenje operatora dodeljivanja (=) umesto operatora poređenja (==)

|  |  |
| --- | --- |
| 1  2  3  4 | $a = 10;  if($a = 5){    echo "Test.";  } |

Izvršavanje ovog koda izazvaće nekoliko posledica izuzetno bitnih po dalje izvršavanje programa, a sve zbog ubačenog operatora dodeljivanja umesto operatora poređenja:

* + promenljiva $a će u uslovu dobiti vrednost 5 koju će zadržati u nastavku programa, što verovatno ne želimo;
  + kada dođe do dodeljivanja vrednosti promenljivoj $a u uslovnom izrazu, ona će imati vrednost 5. A kako smo rekli u prethodnim lekcijama, sve integer vrednosti osim nule, kada se pretvore u boolean tip, postaju true, pa čak i ako su negativne. Jedini slučaj kada će boolean od integera postati false jeste da je taj integer nula. U slučaju broja 5, dobija se true, i tako će ovaj uslov zauvek biti ispunjen, ma šta mi radili ovom programu pre i posle tog uslova.

**Da li će program prijaviti grešku ukoliko napišete „else if" umesto „elseif"?**

 Program neće prijaviti grešku, sintaksno su obe naredbe tačne

 Program će prijaviti grešku jer naredbe nisu sintaksno tačne

**Vežba 1**

***Problem:***

Na stranu dolazi promenljiva $strana. U promenljivoj je dozvoljeno postojanje jedne od dve vrednosti na osnovu kojih će biti učitana strana. Dve vrednosti su „index" i „proizvodi". Ukoliko je vrednost „index" - učitava se strana index.html, ukoliko je vrednost „proizvodi" – učitava se strana proizvodi.html. Ukoliko se nijedna vrednost ne poklapa sa traženom vrednošću, učitava se strana login.html.

***Rešenje:***

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <?php  $page = "index";    if( $page == "index" ){      $page = "index.html";    }elseif( $page == "products" ){      $page = "products.html";    }else{      $page = "login.html";    }  echo $page;  ?> |

Kako bismo rešili ovaj zadatak, najpre definišemo promenljivu koja će predstavljati string koji testiramo. U našem slučaju je to promenljiva $page, koja se nalazi na drugoj liniji koda. Na istoj liniji vršimo i dodelu vrednosti ovoj promenljivoj, koja je inicijalno index. Zatim koristimo if, else if, else strukturu da bismo obradili svaki scenario koji nam je od interesa onako kako je to definisano postavkom zadatka. Najpre postavljamo if i proveravamo da li je $page isto što i „index":

|  |  |
| --- | --- |
| 1 | if( $page == "index" ){ |

Ukoliko je ovaj uslov ispunjen, na strani će biti prikazan tekst: index.html.

Ukoliko ovaj prvi uslov nije ispunjen, potrebno je da proverimo sledeći uslov. Zato koristimo else if:

|  |  |
| --- | --- |
| 1 | elseif( $page == "products" ) |

Na ovaj način proveravamo da li je vrednost promenljive $page isto što i „products".

Ukoliko nijedan od ova dva uslova nije ispunjen, izvršiće se else blok:

|  |  |
| --- | --- |
| 1  2  3  4  5 | else{        $page = "login.html";      } |

i prikazati na strani tekst: login.html, a to je upravo ono što se i traži zadatkom. Time smo ovaj zadatak završili, а ukoliko želite da proverite da li je kod sasvim funkcionalan, pokušajte da menjate inicijalnu vrednost za promenljivu: $page, a potom proveravajte prikaz na strani.

**Vežba 2**

***Problem:***

U sistemu za praćenje vozila postoje četiri statusa vozila: stoji, kreče se, nestao, nepoznat. Ovi statusi označeni su respektivno brojevima 1, 2, 3 i 4.

U aplikaciju ulazi poslednji poznati status vozila, kao i trenutni status vozila u promenljivama $lastStatus i $status.

*$lastStatus = 2;  
$status = 4;*

Potrebno je dodeliti vrednost promenljivoj $statusName. Ovaj naziv sadržaće tekstualnu reprezentaciju statusa vozila (kreće se, stoji, nepoznat...).

Pritom potrebno je ispoštovati sledeće pravilo: ukoliko je poslednje stanje vozila bilo „kreće se", a novi status je „nepoznat", novo stanje mora biti „nestao".

Promenljivu $statusName treba emitovati na izlaz.

***Rešenje:***

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | $lastStatus = 2;  $status = 4;  $statusName = "unknown";    if($status == 1)          $statusName = "stays";  else if($status == 2)          $statusName = "moves";  else if($status == 3)          $statusName = "dissapear";  else if($status == 4)          {          if($lastStatus == 2)          $statusName = "dissapear";          else          $statusName = "unknown";          }  echo $statusName; |

Rešenjе ovog zadatka prati veoma sličnu strukturu kao i prethodni, sa tom razlikom što se u ovoj vežbi traži da uslovna izvršavanja budu postavljena jedno unutar drugog. Postavkom je naznačeno da treba ispoštovati sledeće pravilo: „...ukoliko je poslednje stanje vozila bilo „kreće se", a novi status je „nepoznat", novo stanje mora biti „nestao"... Zato na samom početku koda koji predstavlja rešenje, definišemo promenljive koje će biti korišćene u daljem ispitivanju. Odmah potom postavljamo uslovno izvršavanje definisano proverom za if i tri else if uslova. U svakom od ovih uslova proveravamo $status (1, 2 i 3). Ova tri statusa je potrebno samo proveriti i na osnovu njih odmah definisati vrednost promenljive $statusName. Kako bismo ispoštovali zahtev u postavci za $status kojem je utvrđena vrednost: 4, vršimo novu unutrašnju proveru. U toj proveri vodimo računa o vrednosti poslednjeg statusa, odnosno o vrednosti promenljive: $lastStatus. Ukoliko je poslednji status: 2, promenljiva $statusName će dobiti vrednost „dissapear“, a u suprotnom: „unknown“. Na ovaj način smo ispoštovali sve zahteve zadatka i naše testiranje je završeno.

Ostaje samo još da na strani prikažemo novokreiranu vrednost, i to postižemo linijom:

|  |  |
| --- | --- |
| 1 | echo $statusName; |

**Vežba 3**

***Problem:***

Potrebno je napisati program koji omogućava korisniku da pogađa zamišljeni broj, željeni broj definišete u kodu. Takođe, da bi korisnik znao koliko je blizu odgovora, obavestite ga porukom ukoliko je razlika između pogođenog i zadatog broja manja od 10. Pokušaj se unosi kroz GET parametar „number“.

***Rešenje:***

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | $secretNumber = 765;  if ($\_GET['number'] == $secretNumber )      {          echo ' <p> Congratulations!!! </p> ';      }  elseif( abs($\_GET['number'] - $secretNumber) < 10 )      {          echo ' <p> You are near to the result!!! </p> ';      }  else      {          echo ' <p> Try again!!!  </p> ';      } |

Kao što je navedeno u postavci zadatka, potrebno je koristiti GET parametre, te ćemo zato najpre razmotriti kako definisati GET parametar i kako njegovu vrednosti koristiti u PHP kodu. GET parametri su deo URL-a, te se zato mogu jednostavno dopisati na već postojeći URL. Recimo da se kod za ovaj primer nalazi u fajlu index.php koji se nalazi na sledećoj lokaciji:

C:\wamp\www\test\index.php

U tom slučaju bismo u address baru pretraživača imali URL koji ima strukturu poput sledeće:

<http://localhost/test/index.php>

Da bismo na ovaj URL napakovali GET parametar koji ima naziv „number“ i nosi vrednost 760, na postojeći URL bismo dopisali:

|  |  |
| --- | --- |
| 1 | ?number=760 |

Dakle, sada bi kompletna adresa izgledala ovako:

<http://localhost/test/index.php?number=760>

Ako bismo sada pristupili ovakvoj adresi, PHP bi bio u stanju da prepozna GET parametar pod imenom „number“. Pristupanje ovom parametru bismo postigli sledećom linijom:

|  |  |
| --- | --- |
| 1 | $\_GET['number'] |

Sada možemo preći na sam PHP kod za rešavanje zadatka. Najpre definišemo promenljivu $secretNumber, kojoj dodeljujemo vrednost za broj koji ćemo tražiti (proizvoljan broj).

Odmah nakon ovoga proveravamo da li je korisnik pogodio skriveni broj. To vršimo sledećim kodom:

|  |  |
| --- | --- |
| 1  2  3  4 | if ($\_GET['number'] == $secretNumber )      {          echo ' <p> Congratulations!!! </p> ';      } |

Ukoliko korisnik nije pogodio broj, treba proveriti da li je bar blizu tačnog broja. To postižemo sledećim delom koda:

|  |  |
| --- | --- |
| 1  2  3  4 | elseif( abs($\_GET['number'] - $secretNumber) < 10 )      {          echo ' <p> You are near to the result!!! </p> ';      } |

Kako bismo obezbedili adekvatno obaveštenje i u slučajevima kada je korisnik poslao kroz GET mnogo manji broj od traženog, najpre broj koji je korisnik poslao umanjujemo za vrednost skrivenog broja, a zatim izračunavamo apsolutnu vrednost tog rezultata. Na ovaj način ne možemo u rezultatu dobiti negativan broj. Apsolutna vrednost broja se izračunava funkcijom abs(), kojoj kao parametar prosleđujemo broj čiju apsolutnu vrednost treba izračunati.

Ukoliko nijedan od ova dva uslova nije ispunjen, treba obezbediti alternativni blok koda, koji će se u tom slučaju izvršiti:

|  |  |
| --- | --- |
| 1  2  3  4 | else      {          echo ' <p> Try again!!!  </p> ';      } |

Ovim je naš zadatak završen, i izmenom vrednosti GET parametra „number“ možemo proveriti kako kreirana logika funkcioniše.

**Vežba 4**

***Problem:***

U prodavnici kancelarijskog nameštaja vlasnici su utvrdili akcijske popuste u januaru za kupovinu radnih stolica prema sledećem pravilniku:

* + Za manje od 10 kupljenih stolica – nema popusta,
  + Za 10 – 49 kupljenih stolica – popust od 5%,
  + Za 50 – 99 kupljenih stolica – popust od 10%,
  + Za kupljenih 100 ili više stolica – popust od 15%.
  + Potrebno je napisati programski kod koji će koristeći se if else strukturama izračunavati popuste prilikom kupovine. Preporuka: koristite se operatorom konjunkcije ( &&).

***Rešenje:***

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | $chairs = 10;  if ( $chairs < 10)  {  $discount = 0;  }  elseif ( $chairs >= 10 && $chairs <= 49 )  {  $discount = 5;  }  elseif ( $chairs >= 50 && $chairs <= 99 )  {  $discount = 10;  }  elseif ( $chairs >= 100 )  {  $discount = 15;  }  echo $discount; |

Kao što smo ranije objasnili, u prethodnom kodu korišćena je elseif naredba sa međusobno isključivim uslovima (samo jedan od uslova može biti zadovoljen). Kako znamo da nema provere narednog uslova ukoliko je prethodno ispunjen ovaj primer, možemo uraditi i na sledeći način:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | $chairs = 100;  if ( $chairs < 10)  {  $discount = 0;  }  elseif ( $chairs <= 49 )  {  $discount = 5;  }  elseif ( $chairs <= 99 )  {  $discount = 10;  }  elseif ( $chairs >= 100 )  {  $discount = 15;  }  echo $discount; |